實驗四 STM32 GPIO System

0410137 劉家麟 0416324 胡安鳳

1. 實驗目的

- 了解 STM32 基本輸出入 I/O port 使用原理
- 設計簡易 LED 跑馬燈程式
- 了解按鈕與指撥開關使用原理

1. 實驗原理

請參考上課 GPIO 講義與 STM32L4x6 Reference manual。

2. 實驗步驟

2.1. Lab4.1: LED pattern displayer

參考講義上的教學完成 4 個 GPIO output 初始化，並在麵包板上完成 4 個為 **Active Low** 的 LED 燈電路(當 GPIO 輸出’1’時燈暗，’0’代表燈亮)。

Please Refer to the tutorial on the lecture slide for finishing the initialization of GPIO output and constructing 4 active low LED circuits. (Turn off the LED when GPIO output “1”, and turn on when GPIO output “0”)

Note: LED 需連接至實驗板上的 PB3, PB4, PB5, PB6

Note: Please connect the LEDs to PB3, PB4, PB5, PB6 on board.

完成依以下 Pattern 閃爍的跑馬燈程式。

Please complete the program below and let the LEDs blink as the pattern requirement defined.

2.1.1. Pattern requirement

1 代表 LED 亮，0 代表 LED 暗

“1” represents that LED is on, and “0” represents LED is off.

初始狀態：最右邊的 LED 亮

Initial state: The rightest LED is on.

0	0	0	1

接著**每一秒鐘** LED 依序往左位移，此時會有 2 個 LED 亮

Then the LED shift left in order every one second. At this time, there should be two

LED illuminated.

0	0	1	1

←

0	1	1	0

1	1	0	0

當 LED 亮至最左邊時的下一秒改變位移方向，由左至右

Change the shifting direction to right when the LEDs' state is "1 0 0 0".

1	0	0	0

→

1	1	0	0

當回至初始狀態後在改變位移方向，並重複以上步驟

Change the shifting direction to left when the LEDs' state back to the initial state (0 0 0 1). Repeat the process above.

完成以下程式碼，並利用 leds 這個變數紀錄目前位移數值，與 DisplayLED 函式輸出 leds 數值顯示至 4 個 LED 上。

Please complete the program below and use the variable "leds" to record the LEDs' states. Using function "DisplayLED" to output the "leds" value to the LEDs to display.

Note: 需用位移指令 LSL 或 LSR 進行數值位移

Note: You may need to use LSL or LSR instructions to shift bits.

```
.data
  leds: .byte 0

.text
   .global main

main:
    BL   GPIO_init
   MOVS R1, #1
   LDR  R0, =leds
   STRB R1, [R0]

Loop:
   //TODO: Write the display pattern into leds variable
   BL   DisplayLED
    BL   Delay
   B    Loop

GPIO_init:
  //TODO: Initial LED GPIO pins as output
  BX LR

DisplayLED:

   BX LR

Delay:
    //TODO: Write a delay 1sec function
   BX LR
```

1. GPIO_init

Declare the GPIO port B at text section

```
.text
    .global main  /*Start from manual p75 of GPIO Address
data*/
    .equ RCC_AHB2ENR  , 0x4002104C
    .equ GPIOB_MODER  , 0x48000400
    .equ GPIOB_OTYPER , 0x48000404
    .equ GPIOB_OSPEEDR, 0x48000408
    .equ GPIOB_PUPDR  , 0x4800040C
    .equ GPIOB_ODR    , 0x48000414
    .equ onesec, 800000
```

And use their address to set up the mode in GPIO_init

Reference: http://www.nimblemachines.com/stm32-gpio/

For example, we set PB3 to PB6 input mode.

```
    //enable the port b GPIOB_MODER for output mode, chiech
is 01 (GPOM)
    ldr r0, =GPIOB_MODER
    ldr r1, [r0] //get originally initilized reset value 0xFFFFFEBF
    mov r2, 0x00001540 //0001010101(mode6 to
mode4)000000
    //clear pb6~pb3 to zero
    and r1,r1, 0xFFFFC03F //FFFF1100000000111111 from
manual p25
```

```
	orr r1,r1,r2 //get the value of  FFFF|11|00000000|111111 or 
0000|00|01010101|000000
	str r1,[r0]
```

2. DisplayLED

We initialized LEDS with 0xfff3

```
first_led:
	mov r1, 0xfff3
	strh r1, [r2]
	bx lr
```

Then, run in a loop which divides display leds into switch_left and switch_right.

```
Loop:
	//TODO: Write the display pattern into leds variable
	switch_left:
	mov r3, 0x0
	b goleft
	switch_right:
	mov r3, 0x0
	b goright
	B		Loop
```

In goleft, we do delays and keep left shift LEDs pattern. And do inversely in goright.

```
goleft:
	push {r3}
	ldr r3, =onesec
	bl Delay
	lsl r1, r1, #1
	/*cmp r1, 0xffffff38cmp r1, 
0b11111111111111111111111100111000 //leftboundary*/
	pop {r3}
	cmp r3, #3
	it eq
	moveq r1,0xff3f //special case of shift logic

	strh r1,[r2] //store to output value
```

```
add r3, r3, #1
cmp r3,#4
beq switch_right
bne goleft
```

3. Demo

https://youtu.be/wrXtI7e3BcE

2.2. Lab4.2 Push button

初始化 GPIO PC13 為 Pull-up input，並設計一程式 Polling 實驗板上的 User button 狀態，當 button 按下再放開可以控制 Lab4.1 跑馬燈的停止與啓動(按一次停止再按一次啓動…)。

Please initialize GPIO PC13 as pull-up input and design a program to polling the state of the user button on board. Controlling the scrolling of the LEDs by a click on button (click once to stop scrolling and once more to restart scrolling)

Note: 開發板上的 User button 是連接在 PC13 上，請自行參考講義或 STM32L476 datasheet 完成 GPIOC 初始化。

Note: The user button on board is connected to PC13. Please refer to the lecture slides or STM32L476 datasheet to complete the initialization of GPIOC.

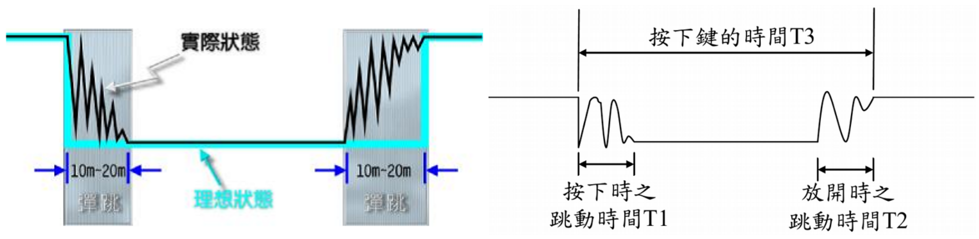
2.2.1. 開關彈跳

使用軟體方式，解決按鍵彈跳問題。

Please solve the button bounce problem using software debounce.

按鍵開關之機械彈跳現象：

按鍵是機械裝置，按壓後，在穩定之前，內部連結會在幾毫秒間來回彈跳。在消除彈跳的這段時間裡，low 和 high 的訊號都會偵測到，造成誤判。



1. **<u>Initialize PC13</u>**

Declaration

```
.equ GPIOC_MODER  , 0x48000800
.equ GPIOC_OTYPER ,   0x48000804
.equ GPIOC_OSPEEDR,   0x48000808
.equ GPIOC_PUPDR  ,   0x4800080c
.equ GPIOC_IDR    , 0x48000810
```

Initialization

```
//enable the port c GPIOC_MODER for input mode
ldr r0, =GPIOC_MODER
ldr r1, [r0]
//clear pc13 to zero
and r1, r1, 0xf3ffffff
```

```
	str r1,   [r0]
```

2. Debouncing

Because there's debouncing problem, we check the button every cycle, and set a threshold for it. If it achieved the threshold, we view it as stable.

```
check_button: //check every cycle, and accumulate 1
	ldr r5, [r4] //fetch the data from button
	lsr r5, r5, #13
	and r5, r5, 0x1 //filter the signal
	cmp r5, #0
	it eq
	addeq r0, r0 ,#1 //accumulate until the threshold

	cmp r5, #1 //not stable, go back to accumulate again
	it eq
	moveq r0, #1

	cmp r0, #1000 //threshold achieved BREAKDOWN!
	it eq
	eoreq r6, r6, #1 //r6^=1

	b check_end
```

3. Demo

https://youtu.be/NvKnooI8_20

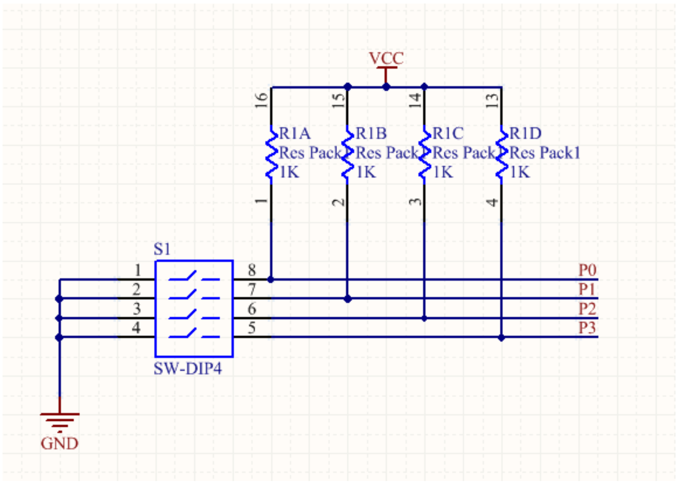
ver 2 with music

https://youtu.be/-l3dshXAFY8

2.3. Lab4.3 密碼鎖

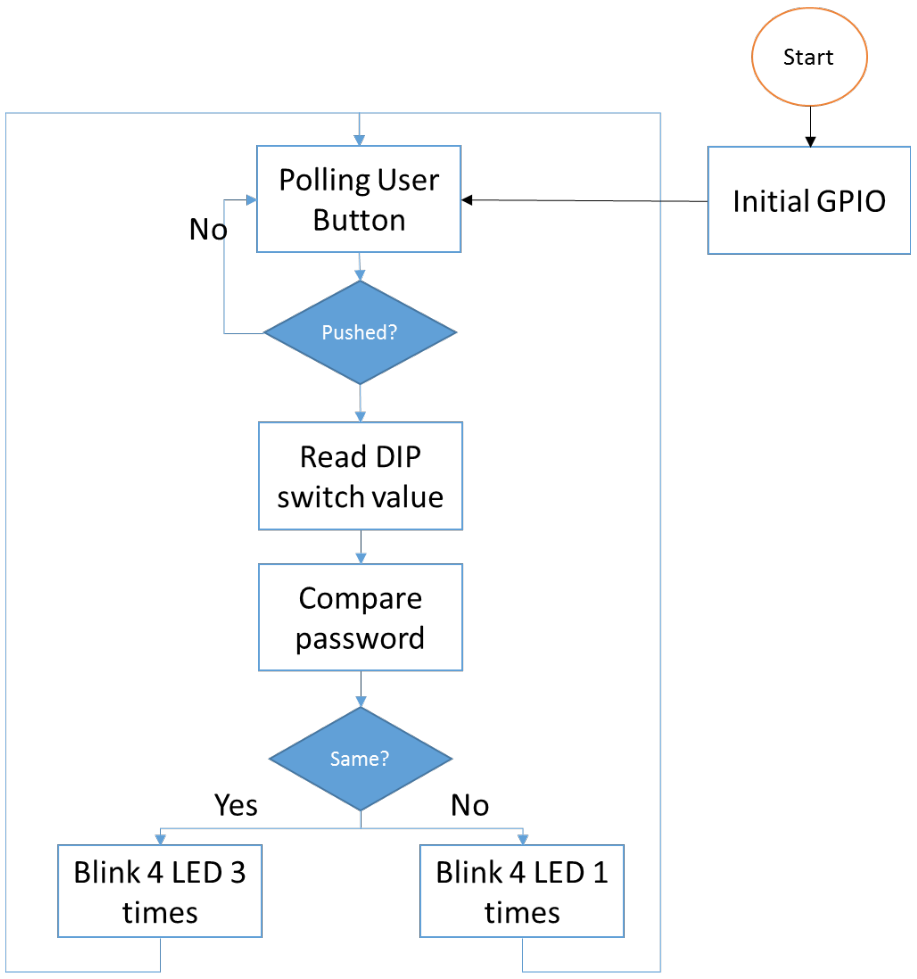
利用麵包板連接 DIP switch 的 active low 電路並連接 P0~P3 至實驗板的 GPIO Pin (同學可自行決定連接的 Pin)

Please use breadboard to construct an active low DIP switch circuit and connect P0~P3 to GPIO pins on board. (You could choose the pins by yourselves)



在程式中宣告一個 password 1byte 全域變數並實做一個簡易的 4bit 密碼鎖程式，其流程如下:

Please declare a 1 byte global variable “password” and implement a simple 4 bits coded lock. Referring to the process below



Note: DIP switch ON 代表’ 1’, OFF 代表’ 0’，若使用者輸入“ON ON OFF OFF”則代表”1100”。Blink 時間間隔 0.5s

Note: Defining DIP switch ON as “1”, OFF as “0”. Thus, when user input “ON ON OFF OFF”, it’s code is “1 1 0 0”. Please set the blink frequency to 0.5s.

1. Debouncing same as problem2, the threshold-accumulation method.

    Loop:

    mov r0, r0 //continue to accumulate the signal of threshold, which is used for debouncing

    b check_button

    check_end:

    cmp r6, #1 //button is pressed, check lock

    beq check_lock

```
blink_end:
mov r6, #0
B           Loop
```

2. Use bitmask and and eor to implement the active low logic, password checking

```
check_lock:
ldr r3, =wait_for_input //if slow motion debug, comment this line
bl delay_quarter_sec //if slow motion debug, comment this line
ldr  r7, [r8]
and  r7, 0b1111
eor  r7, 0b1111
ldr  r9, =password
ldrb r9, [r9]

cmp r7, r9
beq led_blink_three
b led_blink_once
```

3. Demo video https://www.youtube.com/watch?v=cawIGflq5y8

4.More code at:

https://github.com/Alfons0329/MPSLab_Fall_2017/blob/master/Lab4_prob3_ver2/src/main.s